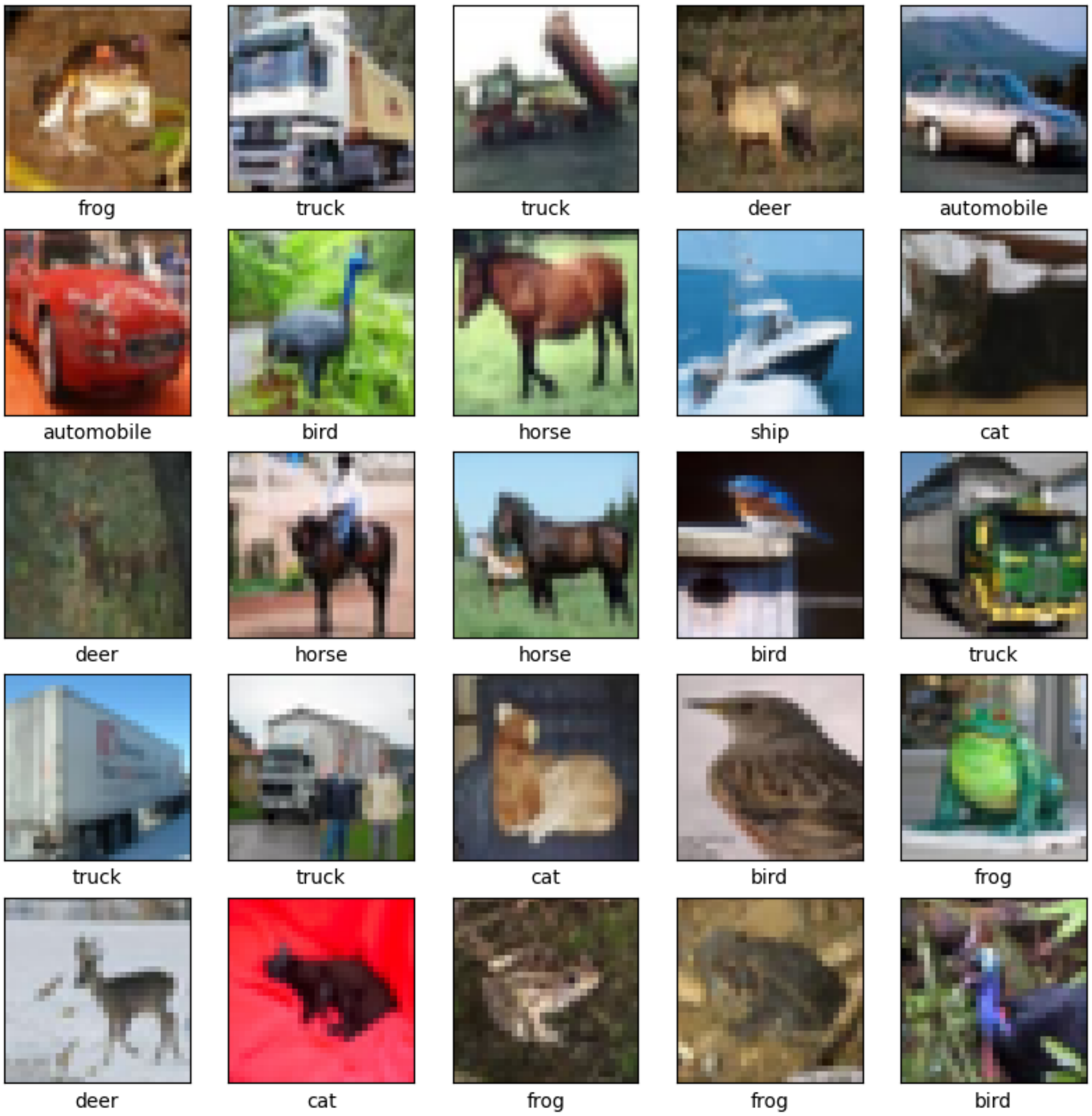
```python
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

# Normalize the pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-
python.tar.gz
170498071/170498071 [==============================] - 5s 0us/step
```

```python
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])

    #First 25 images as 5 by 5 matrix, show images and labelling
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```




```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
#Convolutional base (Height,width color channels)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896
```

```
 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

```python
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
#Flatten 3d output to 1d then place dense layer on top

model.summary()
```

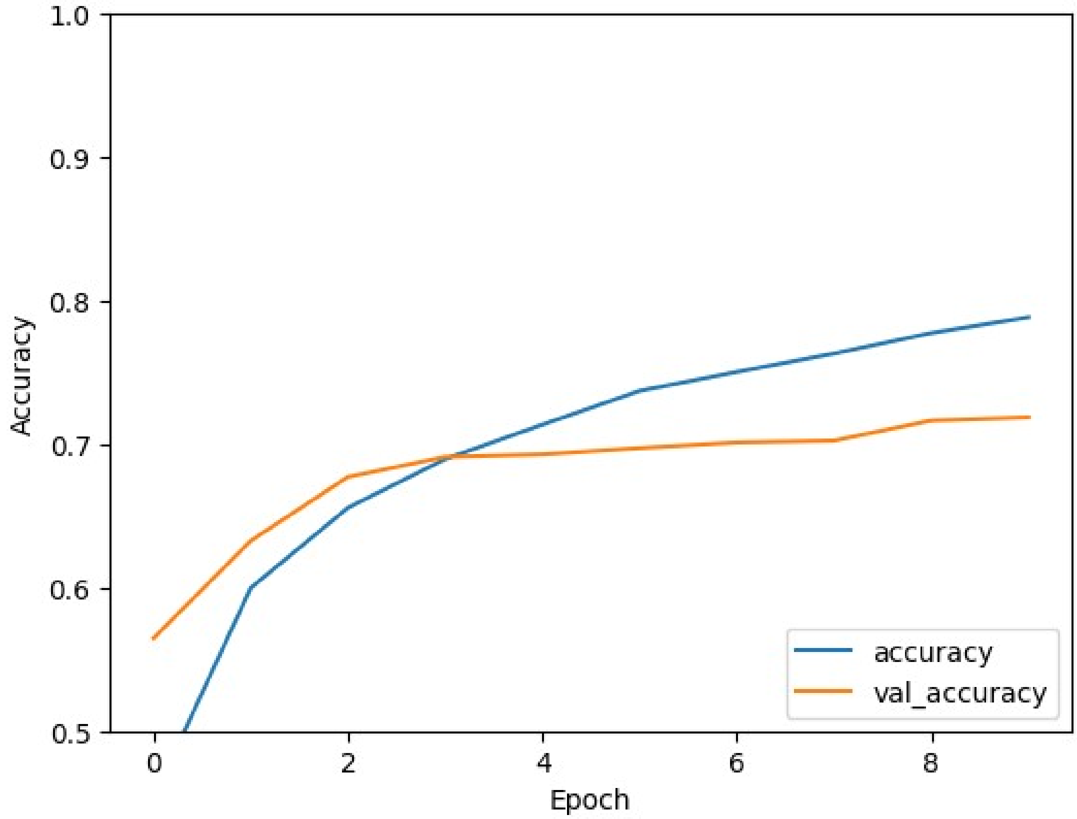
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
#compile and train models
```

```
Epoch 1/10
1563/1563 [==============================] - 89s 56ms/step - loss:
1.5027 - accuracy: 0.4536 - val_loss: 1.2172 - val_accuracy: 0.5648
Epoch 2/10
1563/1563 [==============================] - 85s 54ms/step - loss:
1.1316 - accuracy: 0.5999 - val_loss: 1.0489 - val_accuracy: 0.6327
Epoch 3/10
1563/1563 [==============================] - 85s 54ms/step - loss:
0.9829 - accuracy: 0.6557 - val_loss: 0.9322 - val_accuracy: 0.6772
Epoch 4/10
1563/1563 [==============================] - 87s 56ms/step - loss:
0.8866 - accuracy: 0.6896 - val_loss: 0.8917 - val_accuracy: 0.6911
Epoch 5/10
1563/1563 [==============================] - 93s 60ms/step - loss:
0.8193 - accuracy: 0.7136 - val_loss: 0.8944 - val_accuracy: 0.6929
Epoch 6/10
1563/1563 [==============================] - 91s 58ms/step - loss:
0.7604 - accuracy: 0.7372 - val_loss: 0.8813 - val_accuracy: 0.6973
Epoch 7/10
1563/1563 [==============================] - 81s 52ms/step - loss:
0.7094 - accuracy: 0.7503 - val_loss: 0.8778 - val_accuracy: 0.7012
Epoch 8/10
1563/1563 [==============================] - 82s 52ms/step - loss:
0.6758 - accuracy: 0.7632 - val_loss: 0.8725 - val_accuracy: 0.7025
Epoch 9/10
1563/1563 [==============================] - 84s 54ms/step - loss:
0.6347 - accuracy: 0.7775 - val_loss: 0.8451 - val_accuracy: 0.7165
Epoch 10/10
1563/1563 [==============================] - 83s 53ms/step - loss:
0.6013 - accuracy: 0.7884 - val_loss: 0.8419 - val_accuracy: 0.7187
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)
```

```
313/313 - 5s - loss: 0.8419 - accuracy: 0.7187 - 5s/epoch - 17ms/step
```




```python
print(test_acc)
```

```
0.7186999917030334
```